MARKET-BASED COMPLEX TASK ALLOCATION FOR MULTIROBOT TEAMS

Robert Zlot* and Anthony Stentz
The Robotics Institute, Carnegie Mellon University
Pittsburgh, PA, 15213

ABSTRACT

In order for a team of autonomous robots to perform a complex mission effectively, an efficient assignment of tasks to robots must be determined. Existing multirobot task allocation algorithms treat tasks as simple, indivisible entities. However, when dealing with complex tasks, the structure and semantics of the tasks can be exploited to produce more efficient team plans by giving individual robots the ability to come up with new ways to perform a task, or by allowing multiple robots to cooperate by sharing the subcomponents of a task, or both. In this paper we detail a method for efficiently allocating a set of complex tasks to a robot team. The advantages of explicitly modeling complex tasks during the allocation process is demonstrated by a comparison of our approach with existing task allocation allocation algorithms in an area reconnaissance scenario. An implementation on a team of outdoor robots further validates our approach.

1 INTRODUCTION

Multirobot systems are becoming increasingly more capable and the types of achievable applications for teams of robots are becoming progressively more complex. Many approaches to multirobot coordination rely on a mechanism for task allocation to determine an efficient assignment of tasks to robots; however, existing techniques do not fully consider the complexity of the tasks to be allocated. For the most part, tasks are assumed to be atomic units that can be performed by one or more robots on the team. In practice, tasks are usually introduced to a system by a system or a central planner, and the task allocation algorithm then considers the task only in terms of the level of description provided by either of these sources. However, by reasoning about the task complexity the team can potentially perform the mission more efficiently.

In this paper, we address the problem of allocating *complex* tasks to a team of autonomous robots. Complex tasks are tasks that may have many potential solution strategies; finding a plan to achieve a complex task often involves solving an $\mathcal{NP}$-hard problem. In contrast, *simple* tasks can be executed by a robot in a straightforward, prescriptive manner. In particular, we focus here on complex tasks that can be decomposed into multiple inter-related subtasks (which may also be complex). This class of tasks is natural for describing many application domains, including reconnaissance, automated construction, search and rescue, hazardous waste cleanup, and planetary exploration.

When decomposing a complex task, the most appropriate choice of subtasks in the decomposition is highly dependent on which robot(s) are executing the tasks. While it is possible to first decompose each complex task and then assign the resulting simple tasks, this approach can result in highly suboptimal solutions. One must recognize that there are essentially two problems to solve: *what needs to be done?* (task decomposition); and *who is doing what?* (task allocation). In terms of finding the most efficient solution, it is not possible to determine how best to allocate the tasks if it is not known how they will be decomposed, and it is not possible to determine how best to decompose the tasks if it is not known to whom they will be assigned. In this paper, we address these problems by proposing a distributed algorithm in which robots simultaneously and continuously allocate and decompose complex tasks. Our solution uses a market-based approach in which both efficient task allocations and efficient task decompositions are favored.

In the next section, we review existing approaches to task allocation that can be applied to complex task domains. We then outline our approach in section 3. Results follow from an area reconnaissance scenario implemented both in simulation and on a team of outdoor robots.

2 RELATED WORK

The majority of multirobot systems that utilize an explicit task allocation mechanism assume either that a static set of tasks is given to the system as in-

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **00 DEC 2004**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **Market-Based Complex Task Allocation For Multirobot Teams**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **The Robotics Institute, Carnegie Mellon University Pittsburgh, PA, 15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001736, Proceedings for the Army Science Conference (24th) Held on 29 November - 2 December 2005 in Orlando, Florida. , The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**8**	

put (Botelho and Alami, 1999; Dias, 2004; Gerkey and Matarić, 2002; Golfarelli et al., 1997), or that tasks arrive dynamically, either from external (Dias, 2004; Gerkey and Matarić, 2002) or internal (Simmons et al., 2000a; Zlot et al., 2002) sources. In any case, such approaches search for an efficient assignment of the current task set to robots, assuming that all tasks are indivisible. When this type of mechanism is applied to complex tasks, a robot assigned a task can decompose it and then execute the resulting simple tasks (*e.g.* (Botelho and Alami, 1999)). In reality, however, it may be beneficial to allocate subcomponents of these tasks to multiple robots, and generally the preferred task decomposition will depend on the subtask assignments. Therefore, treating tasks as atomic entities is not always prudent.

A common alternative among systems that explicitly handle complex tasks is a two-stage approach: first decompose all tasks and then distribute the resulting set of subtasks (Aylett and Barnes, 1998; Caloud et al., 1990; Simmons et al., 2000b). The main drawback of this approach is that task decomposition is performed without knowledge of the eventual task allocation; therefore the cost of the final plan cannot be fully considered. Since there is no backtracking, costly mistakes in the central decompositions cannot be rectified. In some instances, the central plan is left intentionally vague, which allows for a limited amount of flexibility in modifying it later. For example, in GOFER (Caloud et al., 1990), the central planner produces a general plan structure for which individual robots can later instantiate some variables; while in the "playbook" system of Simmons *et al.* (Simmons et al., 2000b), the planner relies on a central executive to both allocate tasks and to fill in some plan details.

The M+ cooperative task achievement scheme (Botelho and Alami, 2000) allows some reallocation of subcomponents of complex tasks. Tasks are first allocated using the M+ task allocation mechanism (Botelho and Alami, 1999), which uses a bidding protocol to distribute predefined abstract tasks among the team. Each robot locally decomposes its tasks into *actions* (which can be viewed as primitive subtasks). The task achievement scheme allows the elimination or transfer of actions to remove some inefficiencies or redundancies in the global plan. While this additional step can potentially improve the solution quality to some degree, the initial task allocation does not consider how the actions are to be shared among the robots, nor does the decomposition step consider the eventual allocation of its resulting actions. Additionally, the task achievement scheme does not explicitly model the costs of the actions that get reassigned or canceled.

3 APPROACH

Our approach to the complex task allocation problem builds upon the success of existing market-based multirobot coordination techniques (*e.g.* (Botelho and Alami, 1999; Dias, 2004; Gerkey and Matarić, 2002; Golfarelli et al., 1997; Simmons et al., 2000a; Zlot et al., 2002)). By generalizing the definition of a task and developing appropriate mechanisms to handle these new task descriptions, we create a marketplace capable of distributing complex tasks among a robot team in an efficient manner.

3.1 Market-based Multirobot Coordination

Market-based approaches to multirobot coordination treat a team of robots as participants in a virtual economy, where they can trade contracts for performing tasks or sharing resources in exchange for payment. Cost and revenue functions are designed so that individuals efforts to maximize profits lead to globally efficient solutions.

Our approach can be considered an extension of *TraderBots* (Dias, 2004). In *TraderBots*, agents called *traders* participate in a market, trading tasks via auctions. When an auction is announced, participants compute bids based on their expected profit for the tasks on offer, and the robots that can perform the tasks for the best price are awarded the resulting contracts. Since only profitable trades occur, each auction acts to improve the overall solution. Each *RoboTrader* (a trading agent running on each robot) maintains a schedule of tasks to which it has committed, and can evaluate new tasks by computing the marginal costs of adding them to its schedule. Additionally, *OpTrader* agents can act on behalf of system operators, contracting tasks as they are requested and monitoring the progress of the team. Traders can take on the roles of auctioneer and bidder dynamically, thus facilitating peer-to-peer trades amongst the team. This implies that tasks can be reallocated, allowing for solution improvements over intial assignments and for adapting the task assignments as new information is ascertained. Having no single auctioneer also avoids the presence of a central agent becoming a critical point of failure for the system.

3.2 Task Trees

To introduce complex tasks into a market framework, we use a task tree representation. A task tree is defined as a rooted set of task nodes connected by

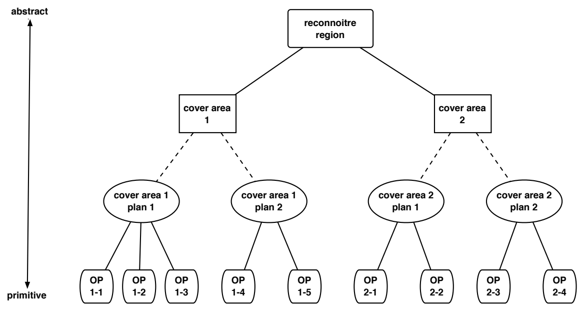

Figure 1: An example AND/OR task tree for an area reconnaissance scenario. The solid edges represent AND operators, and the dashed lines represent OR operators. Note that for the *cover area* tasks, there are two alternative decompositions specified in the tree.

directed edges that specify parent-child relationships between the tasks. Each successive level of the tree represents a further refinement of a complex task; the root is the most abstract description, and the lowest level contains primitive tasks that can be executed by a robot or another agent in the system.

The way in which subtasks are related to their parents can vary depending on the application and the degree of coordination desired. In this paper, we look specifically at a subclass of complex tasks: loosely-coupled tasks related through AND and OR logical operators; to satisfy an AND-labeled task, all of its subtasks must be executed, while an OR task is satisfied when at least one of its subtasks is executed. The AND/OR tree shown in figure 1 represents a task decomposition for an area reconnaissance mission.

3.3 Complex Task Markets

In order to effectively incorporate complex tasks, multirobot task markets can be extended to include *task tree auctions*. Instead of trading contracts for simple tasks, trees of tasks are offered in auctions. Participants can bid on any combination of nodes in the tree, and the auctioneer can choose to award several nodes from the same tree to multiple winning bidders. The winners of the auction are responsible to the seller and must ensure that the tasks are completed before receiving payment (either by executing the task themselves, or by subcontracting parts of the task to other teammates in future negotiations). Because bids can be on tasks at multiple levels of abstraction, task tree markets have the flexibility to allocate tasks at whichever granularity of abstraction is most appropriate.

One benefit of a task tree market is that task tree structures allow robots to express their valuations for both tasks *and* plans. Since robots have different states, information, resources and capabilities, they may prefer different decompositions for the same task. The calculation of these preferences is a recursive process. In the base case, primitive tasks are handled in much the same way as in simple task markets: bids are based on the expected marginal cost of the task, and awards can be immediately inserted into the schedule of the winner. Abstract tasks are evaluated in two ways. First, the cost of performing the task as offered (*i.e.* the auctioneer's decomposition) is considered. This is essentially the cost of performing a set of the descendant primitive tasks; the chosen set depends on the connectives (*e.g.* the logical operators) present in the subtree. In the second step, the bidder can try to find a better plan for the task: it computes its own decomposition of the task, and, if the resulting plan is of lower cost, it uses this cost for its bid. If the bidder is awarded this task, it can insert the primitive subtasks of whichever plan was more beneficial into its schedule. It may later choose to subcontract some or all of the new plan in future auctions.

3.4 Bidding and Auction Clearing

Auction clearing is the process of determining to which bidders to award which tasks. In a task tree auction, the goal of the auctioneer is to find the allocation which maximizes the auctioneer's profit (minimizing team cost). The winning allocation must satisfy several constraints. Firstly, the tree structure defines the tasks that can be awarded in combination. For example, a task node should not be awarded to a bidder in a task tree auction if one of its ancestor nodes has already been assigned. Second, due to the dependency of task costs on previous assignments, only one tree node can be awarded to each bidder.

In each auction, traders are permitted to bid on any arbitrary set of nodes in a task tree (provided they don't represent tasks that are already completed or subcontracted to another party). This gives the bidders a highly expressible format in which to specify their preferences, but creates a difficult auction clearing algorithm. In a previous publication (Zlot and Stentz, 2003), we describe a clearing algorithm for a much simpler bidding language. In that case, participants are allowed to bid on all nodes along a root-to-leaf path in a task tree (a single path starts at the root, but can branch at OR nodes), and the auction clearing problem becomes much easier. The clearing algorithm that solves this problem (described in algorithm 1) is used as a basis for solving the much harder clearing problems that arise when traders can bid on all nodes of the tree.

Algorithm 1: Auction clearing algorithm for restricted bids

```
Let $T$ be a set of nodes in a tree with root $R$. Let $p(N)$, $N \in T$ be the lowest price bid on each tree node;
1 Initialize the solution by marking all leaf nodes;
2 while $N_p \neq R$ do
3     Find $N_{max}$, the maximum depth node in $T$. Let $N_p$ be the parent of $N_{max}$, and $S$ be the set of children of $N_p$;
4     if $Op(N_p) = AND$ then
5         $p(S) := \sum_{C \in S} p(C)$;
6     else if $Op(N_p) = OR$ then
          $p(S) := min_{C \in S} p(C)$;
7     if $p(N_p) < p(S)$ then
          Mark $N_p$, unmark all descendants of $N_p$;
      else
8         $p(N_p) := p(S)$
9     $T := T - S$
```

To clear more general auctions in which participants can bid on any nodes in the tree, two new algorithms are required, depending on the context of the auction.

RoboTrader auctions: This clearing algorithm is used in auctions held by individual robots. The distinguishing property of these auctions is that any tree nodes that aren't sold remain contracted to the auctioneer. That is, the auctioneer has a reserve price for the tree (the cost to perform the tasks itself). A rough outline of this clearing algorithm is given as algorithm 2.

Algorithm 2: *RoboTrader* auction clearing algorithm

```
1 while the tree is not satisfied do
2     Determine and mark the cost minimizing set of bids that satisfy the remaining part of the tree, using algorithm 1;
3     Sort marked nodes by profit (i.e. the difference between the auctioneer's reserve price and the bid price). Retain all bids in that list until we reach a repeat winner. Discard all remaining bids;
```

OpTrader auctions: This algorithm is used in auctions held by the *OpTrader*, an agent acting on behalf of a system operator. When a user introduces a new complex task for the robots to accomplish, the *OpTrader* decomposes the task and holds an auction to allocate the resulting task tree to the robots. In this case, the agent holding the auction cannot perform the tasks itself, therefore there is no reserve price. A sketch of the clearing algorithm is given as algorithm 3. Note that this algorithm does not necessarily award the entire tree – if some of the tree remains unsold, further auctions are called until the tree is satisfied by the combination of awards.

Algorithm 3: *OpTrader* auction clearing algorithm

```
1 Determine and mark the cost minimizing set of bids that satisfy the remaining part of the tree, using algorithm 1;
2 For each bidder, award the node for which it has the highest margin of victory (i.e. the greatest difference between the first and second price bids), if any;
```

4 EXPERIMENTS

We have tested our approach on an area reconnaissance application. In this scenario, a team of robots is tasked with a reconnaissance mission which involves scouting a number of specified named areas of interest (NAI). To cover each NAI the robots select and navigate to a set of observation points (OP) and view the area with range-limited 360° line-of-sight sensors. Because the NAIs may contain enemies, the robots cannot enter them without incurring a large cost. The mission is achieved when the robots visit a sufficient number of OPs to cover a predefined fraction (75%) of each area.

We use a 2.5-dimensional occupancy grid representation of the environment, in which every cell is marked with a height, a cost, and a benefit for viewing it (*i.e.* if it is within an NAI). We use the D* path planning algorithm (Stentz, 1994) to compute navigation costs, and we calculate area visibility from observation points using a point-to-point inter-visibility algorithm.

4.1 Task Decomposition

To construct a task tree for a problem instance, a mission-level reconnaissance task is decomposed into a set of NAIs, and each NAI in turn is decomposed into a set of OPs. An example of a task tree for an area reconnaissance problem with two NAIs is given in figure 1.

Mission-level decomposition: A connected components algorithm is used to determine connected areas within the region that have high viewing benefits. Each separate area (NAI) found becomes a child to the mission-level task, and a bounding-box representation is used to describe the area coverage task.

NAI decomposition: To find the set of OPs from which to view a given NAI, we look at potential OPs[1] and compute the expected revenue (line-of-sight coverage of the NAI from the OP) and the expected cost of traveling to the OP. These quantities are combined as a weighted difference (revenue minus weighted cost), and we repeatedly choose the highest scoring OP until we have sufficient coverage of the area. By modifying the weighting factor, we can come up with alternative decompositions for the same area. With a high weight, we put more emphasis on cost, and the resulting OP set has low cost for a single robot. Using a low weight deemphasizes navigation cost and results in more spread out, but potentially less, coverage points. These decompositions often favor team plans – the robot performing the decomposition often contracts out some of the goal points to other robots. Our implementation computes one 'individual' decomposition and one 'team' decomposition, which are both placed in the tree as alternative plans under an OR node (as in figure 1). In general, if the team is heterogeneous, robots can come up with several alternative plans that draw on the various capabilities of the different types of robots on the team. NAI tasks that have previously sold or executed OP children can still be decomposed and traded by considering those OPs as being fixed in the decomposition.

4.2 Task Clustering

In addition to the mission and area decomposition steps, a clustering algorithm is also run on the observation points and NAIs. The purpose of this algorithm is to group together any sibling nodes that are physically close together. This can improve the allocation efficiency by allowing some groups of synergistic tasks to be sold together, thus circumventing some local minima. With the addition of task clustering, a task tree for an area reconnaissance mission can have a depth of up to six levels.

4.3 Costing

The goal of the robot team is to complete the mission while minimizing the total distance traveled. Individually, the robots must solve instances of the traveling salesman path problem (TSPP)[2] when deciding in which order to visit observation points. Robots frequently encounter TSPP-related optimization and cost estimation problems when computing reserve prices for auctions, when bidding, and when reordering schedules after trades. For small problems (at most twelve cities) we compute the optimal solution; while for larger instances we run a $\frac{3}{2}$-approximation algorithm (Hoogeveen, 1991), which we then improve by a 2-opt local search. Reserve prices and bids for tasks are computed by differencing the costs of a path with and without the tasks under consideration.

4.4 Simulation Experiments

A series of experiments was conducted to evaluate the effectiveness of task tree trading for the area reconnaissance scenario, using a multirobot simulator with a graphical display (figure 2). In each test, a number of robots and NAIs are randomly placed within a 200x200-cell grid containing multiple obstacles. The terrain map was constructed from real-world measurements by an autonomous helicopter equipped with a downward looking scanning laser rangefinder[3]. The NAIs are non-overlapping, randomly-sized rectangles with edge lengths drawn uniformly at random in the range of 15 to 30 grid cells.

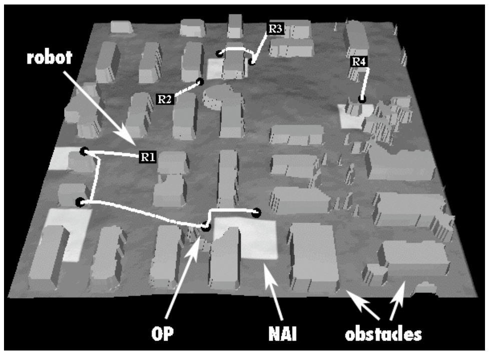

Figure 2: Screenshot from the multirobot simulator. Here there are four robots tasked with a reconnaissance mission requiring the coverage of five areas. Also pictured are the paths that the robots plan to traverse to visit the OPs.

Our experiments examine the benefit of explicitly handling complex task allocation by comparing task tree auctions against "single-level" task auctions, meant to represent existing task allocation algorithms that model tasks as atomic entities and do not consider the structure or complexity of the tasks. In addition to task tree auctions, we look at three particular types of single-level allocation mechanisms.

[1]Because the computation required to consider a single OP is expensive, we limit the candidate OPs to twelve per NAI, which surround the bounding box with four OP candidates per side.

[2]The traveling salesman path problem is similar to the traveling salesman problem, except the salesman is required to follow a path, rather than a tour (*i.e.* the salesman does not have to return to the starting city).

[3]The data is courtesy of Omead Amidi and Ryan Miller.

Goal point-level allocation: Here the mission is initially decomposed by the *OpTrader* into a set of observation points. All auctions that follow are for tasks in this set of goal points only (*i.e.* there is no notion of the NAIs; no further decompositions occur).

Area-level allocation: The *OpTrader* decomposes the mission into a list of NAIs. Subsequent auctions are for these area coverage tasks only. When a *RoboTrader* bids on a task, it can compute its own decomposition but it can never reallocate any of the subtasks (thus there can be no cooperation between robots in covering a single NAI).

Mission-level allocation: The mission is not pre-decomposed: the auctions are for the mission task as a whole. *RoboTraders* can decompose the mission locally, but cannot reallocate subcomponents. This means that in any solution only one robot is executing the mission.

Task tree allocation: Tasks are represented as task trees, and are traded at multiple levels of abstraction. Task tree auctions are expected to outperform area (and mission-level) auctions because the use of task tree auctions makes it possible for traders to share coverage of NAIs when such an allocation is beneficial. In addition, task tree auctions are expected to be superior to goal point auctions because they permit the traders to redecompose the complex tasks and come up with more efficient plans.

There are generally two types of task allocation mechanisms: centralized, and distributed. In centralized allocation, a single agent is responsible for assigning the tasks to the robots. With distributed allocation algorithms, the robots have an initial allocation (possibly coming from an centralized allocation phase) and can then reallocate tasks by having peer-to-peer negotiations. In our experiments we consider both of these settings.

Centralized auctions: The *OpTrader* holds a series of auctions to allocate the tasks to the robots. In the case of task tree auctions, algorithm 3 of section 3.4 is used. For single-level auctions, we use a greedy clearing algorithm, similar to one used in *TraderBots*: the robots send bids for all available tasks to the *OpTrader*; the *OpTrader* then greedily selects the best bids and awards those tasks to the robots (one task per robot); repeat until all tasks have been awarded.

Distributed (peer-to-peer) auctions: The initial allocation is obtained via a centralized auction (as described above). The *RoboTraders* then hold further auctions in rounds, in which each trader, sequentially and in random order, calls and clears one auction. Rounds are held repeatedly until a stable solution is reached (*i.e.* no more awards are being given out). For peer-to-peer task tree auctions, each robot randomly chooses one of the trees to which it has committed for auction, and algorithm 2 is used to clear. For single-level auctions, each *RoboTrader* offers all of its available tasks; the other robots submit bids and a single award is given out for the one task that results in the greatest profit.

The majority of existing multirobot task allocation algorithms fall roughly into the single-level centralized allocation category (Botelho and Alami, 1999; Caloud et al., 1990; Gerkey and Matarić, 2002; Simmons et al., 2000a; Simmons et al., 2000b), although a small number of approaches use some form of single-level distributed allocation (*e.g.* (Golfarelli et al., 1997; Sandholm, 1993)). *TraderBots* (Dias, 2004), and, to some extent, M+ (Botelho and Alami, 1999; Botelho and Alami, 2000) include both single-level centralized and single-level distributed task allocation.

4.5 Results

For each type of allocation mechanism (centralized, distributed) we ran the three single-level allocation algorithms and compared the solutions with that produced by the task tree auction mechanism. To determine how the algorithms are affected by problem complexity, in a first set of experiments we vary the number of areas (from 1 to 10), and in a second set we vary the number of robots (between 2 and 10).

The quality of a solution is quantified as the total distance traveled by the team. In order to evaluate each single-level approach for a given problem instance, we look at the ratio of the solution cost of that approach to the task tree solution cost (*i.e.* a result greater than one indicates that the task tree solution was superior). Results are shown in figures 3-6. In the plots, each data point is averaged over forty trials, and 95% confidence intervals are displayed as error bars.

From the results in figures 3-6, we can see that the task tree allocation mechanism outperforms all of the single-level task allocation algorithms. Furthermore, in most cases, adding peer-to-peer auctions following the centralized auctions results in even greater improvement of the task tree algorithm over the others. Having the flexibility both to replan and to cooperate on complex tasks gives the task tree algorithm an advantage over all of the single-level approaches.

As expected, in the peer-to-peer allocation sce-

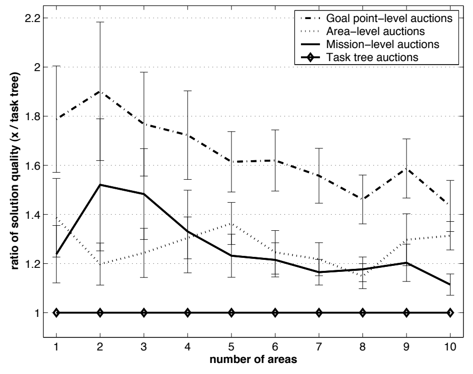

Figure 3: A comparison of solution quality of task tree auctions vs. single-level auctions varying the number of areas. The number of robots is held fixed at five. All auctions are centralized (*OpTrader*) auctions.

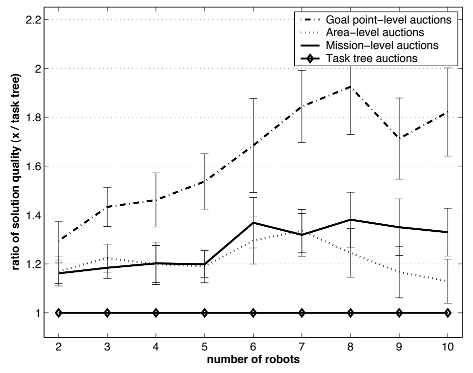

Figure 4: A comparison of solution quality of task tree auctions vs. single-level auctions varying the number of robots. The number of areas is held fixed at five. All auctions are centralized (*OpTrader*) auctions.

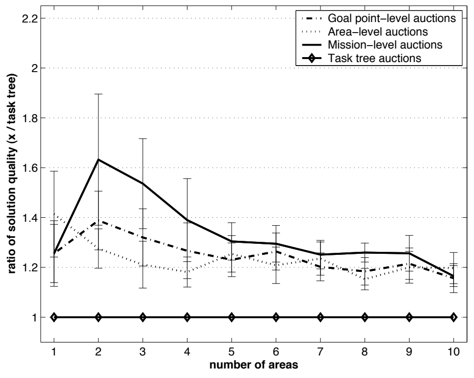

Figure 5: A comparison of solution quality of task tree auctions vs. single-level auctions varying the number of areas. The number of robots is held fixed at five. Centralized (*OpTrader*) auctions are followed by reallocation through peer-to-peer (*RoboTrader*) auctions.

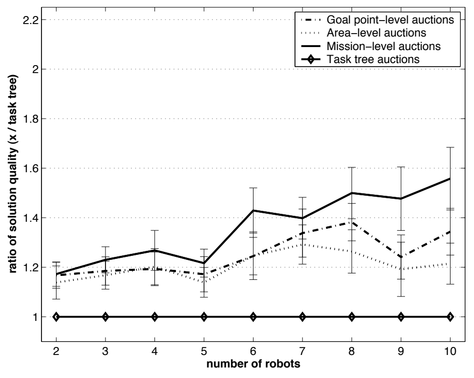

Figure 6: A comparison of solution quality of task tree auctions vs. single-level auctions varying the number of robots. The number of areas is held fixed at five. Centralized (*OpTrader*) auctions are followed by reallocation through peer-to-peer (*RoboTrader*) auctions.

narios, the mission-level allocation scheme produces the worst results. This is due to the fact that the mission-level algorithm assigns the entire mission to one robot, and does not take advantage of the presence of multiple robots. Perhaps surprisingly, in the centralized auctions, mission-level allocation is often better than allocations at other levels; another general trend in the centralized algorithm results, is that area-level allocations always perform better than goal point-level allocations. Both of these effects demonstrate a weakness of greedy single-task allocation algorithms that are typically used in other systems – that is, because each robot is awarded one task in every auction round (excepting the last), bad allocations occur when some of those assignments would have been better-suited for other robots. This is more pronounced when there are more tasks to assign, as is the case with the more primitive-level allocations. As evidenced in figures 5 and 6, peer-to-peer reallocations (as used in *TraderBots*) tend to repair many of these bad allocations.

4.6 Robot experiments

An implementation on a robot platform was also tested on a team of autonomous E-Gators. The E-Gators are outdoor electric utility vehicles manufactured by John Deere, which we fitted with computers and sensors including a tilting laser range scanner, GPS, and gyroscopes. The software architecture is similar to one we have used on a team of Pioneer robots (Dias et al., 2004), and consists of a *RoboTrader* process that communicates with lower-level processes responsible for executing tasks, robot control, interrobot communictions, and data management and mapping. Two modified E-Gators are shown in figure 7.

Figure 8 is a map created by a two E-Gator team tasked with an area reconnaissance mission consisting of three NAIs. The experiments are carried out on a grassy field with some sparse obstacles, such as trees

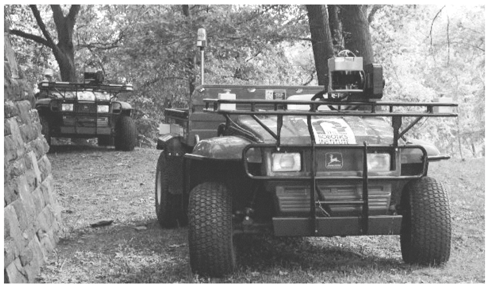

Figure 7: The autonomous E-Gator platform.

and trash bins. The robots initially acquire tasks by task tree-trading with the *OpTrader*, and subsequently hold peer-to-peer task tree auctions with one another. In the final allocation, one robot handles two NAIs and the other one NAI, using their own task decompositions to determine a set of OPs for each area.

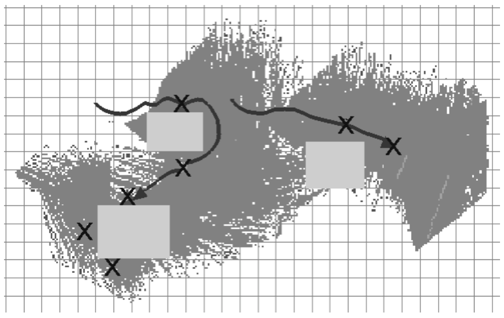

Figure 8: Snapshot of a map created by two E-Gators on an area reconnaissance mission. Three NAIs are marked as lightly shaded rectangles, and seven OPs are shown as black X's. The dark-shaded region is space mapped out by the laser rangefinders. Robot paths are also displayed. Grid cells are $2m$x$2m$.

5 CONCLUSIONS

In this paper, we identify a previously unexplored problem in multirobot task allocation: that is, the allocation of complex tasks. To address this problem, we introduce a market-based solution that uses novel task tree auctions. Empirical evidence from an area reconnaissance scenario demonstrates that our approach improves upon what we call "single-level allocation" algorithms, the current state of the art in multirobot task allocation. An implementation on a team of outdoor robots verifies the feasibility of our approach.

ACKNOWLEDGMENTS

This work was sponsored by the U.S. Army Research Laboratory, under contract **Robotics Collaborative Technology Alliance** (contract number DAAD19-01-2-0012). The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The authors would like to thank Bernardine Dias, Marc Zinck, Boris Sofman, Juan Pablo Gonzalez, Joseph Carsten, and Herman Herman for their important contributions in developing the robot hardware and software, and assisting with field testing.

References

Aylett, R. and Barnes, D.: 1998, A multi-robot architecture for planetary rovers. in *Proceedings of the 5th ESA Workshop on Advanced Space Technologies for Robotics and Automation*

Botelho, S. S. C. and Alami, R.: 1999, M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement. in *Proceedings of the International Conference on Robotics and Automation*

Botelho, S. S. C. and Alami, R.: 2000, Robots that cooperatively enhance their plans. in *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems (DARS)*

Caloud, P., Choi, W., Latombe, J.-C., Pape, C. L., and Yim, M.: 1990, Indoor automation with many mobile robots. in *Proceedings of the International Workshop on Intelligent Robotics and Systems (IROS)*

Dias, M. B.: 2004, *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments. Ph.D. thesis*, Robotics Institute, Carnegie Mellon University

Dias, M. B., Zlot, R., Zinck, M., Gonzalez, J. P., and Stentz, A.: 2004, A versatile implementation of the *TraderBots* approach to multirobot coordination. in *the 8th International Conference on Intelligent Autonomous Systems (IAS-8)*

Gerkey, B. P. and Matarić, M. J.: 2002, Sold!: Auction methods for multi-robot control. *IEEE Transactions on Robotics and Automation Special Issue on Multi-Robot Systems* 18(5)

Golfarelli, M., Maio, D., and Rizzi, S.: 1997, A Task-Swap Negotiation Protocol Based on the Contract Net Paradigm, *Technical Report 005-97, CSITE, University of Bologna*

Hoogeveen, J. A.: 1991, Analysis of christofides' heuristic: Some paths are more difficult than cycles. *Operations Research Letters* 10

Sandholm, T.: 1993, An implementation of the contract net protocol based on marginal cost calculations. in *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*

Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Thrun, S., and Younes, H.: 2000a, Coordination for multi-robot exploration and mapping. in *Proceedings of the National Conference on Artificial Intelligence*

Simmons, R., Apfelbaum, D., Fox, D., Goldman, R. P., Haigh, K. Z., Musliner, D. J., Pelican, M., and Thrun, S.: 2000b, Coordinated deployment of multiple heterogeneous robots. in *Proceedings of the Conference on Intelligent Robotics and Systems (IROS)*

Stentz, A.: 1994, Optimal and efficient path planning for partially-known environments. in *Proceedings of the International Conference on Robotics and Automation*, Vol. 4, IEEE

Zlot, R. and Stentz, A.: 2003, Multirobot control using task abstraction in a market framework. in *Collaborative Technology Alliances Conference*

Zlot, R., Stentz, A., Dias, M. B., and Thayer, S.: 2002, Multi-robot exploration controlled by a market economy. in *Proceedings of the International Conference on Robotics and Automation*